Improved polytope volume calculations based on Hamiltonian Monte Carlo with boundary reflections and sweet arithmetics

Augustin Chevallier, Sylvain Pion, Frédéric Cazals

Improved polytope volume calculations based on Hamiltonian Monte Carlo with boundary reflections and sweet arithmetics

Augustin Chevallier* and Sylvain Pion† and Frédéric Cazals‡

May 11, 2021

Abstract

Computing the volume of a high dimensional polytope is a fundamental problem in geometry, also connected to the calculation of densities of states in statistical physics, and a central building block of such algorithms is the method used to sample a target probability distribution.

This paper studies Hamiltonian Monte Carlo (HMC) with reflections on the boundary of a domain, providing an enhanced alternative to Hit-and-run (HAR) to sample a target distribution restricted to the polytope. We make three contributions. First, we provide a convergence bound, paving the way to more precise mixing time analysis. Second, we present a robust implementation based on multi-precision arithmetic, a mandatory ingredient to guarantee exact predicates and robust constructions. We however allow controlled failures to happen, introducing the *Sweeten Exact Geometric Computing* (SEGC) paradigm. Third, we use our HMC random walk to perform H-polytope volume calculations, using it as an alternative to HAR within the volume algorithm by Cousins and Vempala. The systematic tests conducted up to dimension $n = 100$ on the cube, the isotropic and the standard simplex show that HMC significantly outperforms HAR both in terms of accuracy and running time. Additional tests show that calculations may be handled up to dimension $n = 500$. These tests also establish that multiprecision is mandatory to avoid exits from the polytope.

Keywords. polytope volume, randomized algorithms, multi-phase Monte Carlo, Hamiltonian Monte Carlo, numerics, multi-precision arithmetic, exact predicates.

*Inria, France and Lancaster University, UK, chevallier.augustin@gmail.com

†Inria, France, Sylvain.Pion@inria.fr

‡Inria and Université Côte d'Azur, France, Frederic.Cazals@inria.fr

1 Introduction

1.1 Polytope volume calculations and related problems

Volume calculations. Computing the volume of a polytope – a bounded region of $\mathbb{R}^n$ defined by the intersection of a fixed set of half spaces (H-polytope) or the convex hull of vertices (V-polytope), is a classical problem in science and engineering. Complexity-wise, the problem is #-P hard irrespective of the representation of the polytope (H-polytope or V-polytope) [1]. This observation naturally calls for approximation algorithms [2, 3] delivering (ε, δ) approximations. Over the years, the complexity of volume calculation algorithms, measured by the number of calls to the oracle stating whether a point is inside the polytope, has been lowered from $O^\star(n^{23})$ [4] to $O^\star(n^4)$ [5], and $O^\star(n^3)$ [6], the latter for well rounded bodies. More recently, the complexity $O^\star(hn^{2/3}hn^{\omega-1})$ has been established [7], with h the number of hyperplanes and ω the matrix multiplication exponent. The reader is referred to [7] for the full history. Interestingly, recent volume calculation algorithms are of the *multi-phase Monte-Carlo* type, an iterative strategy where each step benefits from the *progress* made at the previous step. Sketchily, the volume calculation boils down to estimating ratios in a telescoping product, each ratio being the integral over the convex of functions carefully chosen according to a *cooling schedule*. In recent algorithms, exponential functions are used [6, 8]. We also note that recent work has focused on the reduction of the cooling schedule size, using statistical tests to bound the aforementioned successive ratios [9]. The complexity of the algorithm therefore depends on (i) the number of functions in the cooling schedule, (ii) the number of points sampled at each step, and (iii) the complexity of generating a sample according to the specified distribution.

1.2 Sampling a target distribution in a bounded domain

Volume calculations require an algorithm to sample a target distribution π in a bounded domain. We describe such algorithms, providing mixing times for the uniform distribution– for the sake of conciseness–from a warm start.

General idea: MCMC. In non trivial cases, the default strategy is to build a Markov Chain leaving the target distribution π invariant. Under mild additional conditions, computing integrals of a function with respect to the target distribution can be approximated by averaging the function on the samples [10]. This method is called Markov Chain Monte Carlo (MCMC). The convergence of an MCMC schemed is usually assessed by mixing properties, e.g. based on how fast the Markov chains converges to its stationary distribution [11].

Hit-and-run and ball walk. In the case of polytopes, two important Markov chains have been introduced in the literature: Hit and Run (HAR) [12, 13, 14, 15] and Ball Walk [16]. These two Markov chains uses different strategies to stay inside the polytope: ball walk samples within a ball and rejects points outside of the polytope, while Hit and Run only proposes inside the polytope–no rejection step needed. Furthermore, HAR is amenable to several optimizations [17, 18], including the choice of the random line used, and the

calculation of the facet of the polytope intersected by a line. In the context of polytope volume computation, upper bound for the mixing times of the uniform distribution from a warm start exists: HAR mixes in $O^\star(n^3)$ steps while ball-walk mixes in $O^\star(n^{2.5})$ steps [7, 14, 19].

Hamiltonian Monte Carlo. An efficient sampler is Hamiltonian Monte Carlo (HMC) [20, 21]. HMC relies on the measure preserving properties of Hamiltonian flows on phase space to build a Markov Chain leaving π invariant. In a nutshell, a HMC works by adding a velocity/momentum and an HMC step involves three sub-steps which are (i) picking a random velocity p, (ii) following the Hamiltonian flow associated to $H(q,p) = \nabla \log \pi(q) + \frac{1}{2}\|p\|^2$ for a fixed time, and (iii) projecting down in position space. As seen from Hamilton's equation, the fact that the gradient of the target density is used to twist the momentum p rather than the position q helps forcing the dynamical system to *glide* across the typical set [20], which is useful to deal with concentration phenomena. HMC generally uses a Metropolis step when a numerical integration is required for the flow of the Hamiltonian. In our case, we bypass this difficulty by using analytical trajectories.

Riemannian HMC. RHMC uses a Hamiltonian exploiting a metric defined on the domain of interest [22]. For polytopes, a metric based on the Hessian of the barrier function ($\phi(x) = -\sum_{i=1,\dots,d} \log(a_i{}^\mathsf{T} x - b_i)$) can be used to constrain the random walk (RW) within the polytope [23]. This strategy has been used to sample polytopes and compute their volume [7], which yields $O^\star(hn^{2/3})$ as mixing time in the context of polytope volume computation, with h the number of hyperplanes [7]. However, the Hessian of ϕ is ill-conditioned near barriers [24], and we are not aware of any implementation for polytope volume calculations.

Dynamical billiard. Random walks in polytopes are also connected to billiards. In two dimensions, it has been proven that there is a G_δ dense subset of ergodic billiards in the set of all possible billiards [25]. For those billiards, Birkhoff theorem implies that almost every trajectory is uniformly distributed on the phase space of the polygon. While analogous properties are unknown for billiards in dimension $n > 2$ [26], it has been conjectured that billiard trajectories provide valuable building blocks for random walks sampling uniform distributions.

Billiard walk and billiard HMC. A step of billiard walk consists of choosing a direction at random, and following the corresponding billiard trajectory for a fixed time. Properties of billiard trajectories, including their ability to escape from corners, have motivated the sampling algorithm from [27] for general n-dimensional domains, with the velocity refresh ensuring ergodicity. This work is however limited to uniform distributions. In the case of polytopes, billiard walk can be seen as a special case of HMC with reflections on boundaries [28], a strategy used in Bayesian statistics to restrict the state space. We prove the uniform ergodicity of billiard HMC (Thm 3, Sect. 2), and experimentally assess its efficiency for polytope volume estimation (Sections 3 and 4).

1.3 Robustness issues and the SEGC paradigm

It has long been known that geometric algorithms are prone to (almost) degenerate situations, which manifest even on the simplest expressions in 2D [29]. The design of robust geometric algorithms can be done in a general way using the Exact Geometric Computation paradigm [30], as it is done for example in the `CGAL` [31] software library. In order to do so, the `CGAL` kernel distinguishes between predicates and constructions. However, the EGC paradigm faces limitations of theoretical and practical nature. The former refer to the impossibility to provably determine the sign (positive, negative or zero) of some real functions in finite time in all cases. The latter relate to the added complexity which can cause the use of resources like processing time and memory space to hit limits.

Without compromising on the robustness properties of the EGC paradigm, we simply allow controlled failures by allowing the sign functions to return a fourth possible value meaning *I can't compute*. This generalization can practically be implemented using (C++) exception handling mechanisms. We name this extended paradigm Sweeten Exact Computing Paradigm (SEGC). And we refer to *sweet arithmetics* to qualify the kind of arithmetic functions libraries complying with it. In our implementation, we use the `iRRAM` C++ library [32], which provides iterative multiple precision computations for many real functions. This library does not fit into the original EGC paradigm definition, as the zero determination cannot be decided using algebraic separation bounds.

1.4 Contributions

This paper makes contributions touching upon random walks for MCMC algorithms, polytope volume calculations, and Hamiltonian Monte Carlo. More precisely:

1. In section 2, we present a robust version of billiard HMC, and prove its uniform ergodicity for convex bodies.

2. In section 3, we instantiate our random walk to sample distributions used for H-polytope volume calculations. Exploiting analytical expressions for HMC trajectories, we provide a robust implementation of the random walk based on multi-precision interval arithmetic.

3. Finally, section 4 reports experiments comparing HAR and our random walk. The first test samples a target distribution. The second one embeds our random walk into the practical polytope volume calculation of Cousins and Vempala [33]. In both cases, we show superior performances over HAR, for dimension up to $n = 500$.

It is important to note that Riemannian HMC [7] forces the random walk to stay within the domain of interest. This is achieved via a distorted metric based on the barrier function, which also yields ill conditioned numerics. We instead use billiard HMC, and control that the RW remains in the polytope using exact predicates based on multiprecision.

Notations. To conform with previous work, the following notations are used in this paper: (i) ε: criterion used to assess the quality of the volume approximation [33]; (ii) ϵ: notation used for the total variation bound on the mixing time. See theorems 3 and 4. The total variation norm is denoted $\|\cdot\|_{TV}$.

Proofs and pseudo-code. The reader is referred to the appendix.

2 Billiard Hamiltonian Monte Carlo

Consider a bounded open set $Q \subset \mathbb{R}^n$ with piecewise smooth boundary and a target probability measure with density $\pi : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ such that $\pi(x) = 0$ for all x not in the closure $\overline{Q}$ of Q.

2.1 Billiard HMC

Denoting $q^{(i)}$ and $p^{(i)}$ the i-th coordinates of position and momentum respectively, recall Hamilton's equations which state that the velocity field in phase space is orthogonal to the gradient of the Hamiltonian H:

$$\frac{dq^{(i)}}{dt} = \frac{\partial H}{\partial p^{(i)}}, \frac{dp^{(i)}}{dt} = -\frac{\partial H}{\partial q^{(i)}}. \tag{1}$$

As with HMC, we define a potential energy $U(q) = -\log(\pi(q))$ and the Hamiltonian $H(q,p) = U(q) + \frac{1}{2}\|p\|^2$ but this time restricted to $\Gamma = \overline{Q} \times \mathbb{R}^n$. We assume that π is the restriction to $\overline{Q}$ of positive smooth function defined on $\mathbb{R}^n$ and we use Φ_t the Hamiltonian flow. However, the trajectories of this flow are not included in $\overline{Q}$ even if the initial point (q,p) is in $Q \times \mathbb{R}^n$. For every $q \in Q$ and every $p \in \mathbb{R}^n$, we define $T(q,p)$ as the largest T such that for all $0 \leq t < T$, $\Phi_t(q,p) \in Q$. We also define $T(q,p) = 0$ when q is in the boundary of Q.

Following [28] and [34, 27], we modify the flow by forcing reflections on the boundary of Q. This flow, illustrated on Fig. 1, is denoted as follows:

$$\begin{cases} \tilde{\Phi}_t : \overline{Q} \times \mathbb{R}^n \to \overline{Q} \times \mathbb{R}^n \\ (q,p) \mapsto \tilde{\Phi}_t(q,p) \equiv (\tilde{\Phi}_t^{(q)}(q,p), \tilde{\Phi}_t^{(p)}(q,p)). \end{cases} \tag{2}$$

Note that the latter equation defines position and velocity upon applying the flow. However, as noted in [34, 27], this new flow might exhibit problematic trajectories: some of them might not be defined for all t because of singularities on the boundary, others might have huge number of reflections or even an infinite number of reflections in finite time. This does not happen in 2D [35, Section 2.4], but we are not aware of any proof when $n > 2$ or the trajectories are curved. Hence, we introduce the upper bound $M > 0$ for the maximum number of reflections, and cull problematic trajectories accordingly.

Remark 1. *For $q \in Q$ and any p, $T(q,p) > 0$ and $\tilde{\Phi}_{T(q,p)}(q,p)$ is in the boundary of Q.*

Remark 2. *When q is in the boundary of Q and $t > 0$, $\tilde{\Phi}_t(q,p)$ is only defined for the momenta p such that the open half-line with origin q and direction p, is included in Q in a neighborhood of q.*

Algorithm. Let $M \in \mathbb{N}^*$ be the maximum number of reflections allowed. Given a point $q^{(t)} \in Q$, the algorithm is as follow (Algorithm S1):

Algorithm 1 Hamiltonian Monte Carlo with reflections

- (Step 1) Choose the traveling time $L \sim \text{unif}(0,1)$
- (Step 2) Pick the momentum $p \sim \mathcal{N}(0, I_n)$
- (Step 3) If the flow $\tilde{\Phi}_L(q^{(t)}, p)$ is defined and does not involve more than M reflections between $t = 0$ and L, and if $\tilde{\Phi}_L(q^{(t)}, p) \in Q$
  - Take $q^{(t+1)} = \tilde{\Phi}_L^{(q)}(q^{(t)}, p)$
  - Else, take $q^{(t+1)} = q^{(t)}$

For a fixed $L > 0$, steps from 2. and 3. define a Markov kernel $P_{\pi,L}$. The full algorithm (steps from 1. to 3.) define a Markov kernel P_π that can be expressed with $P_{\pi,L}$.

For $L > 0$, let Γ_L be the largest subset of the phase space $\Gamma = \overline{Q} \times \mathbb{R}^n$ where $\tilde{\Phi}_L$ is defined; this set admits no more than M reflections, and the trajectory does not finish in a singularity (a point of the boundary where the normal is not defined) at time L. Γ_L is open and therefore measurable. Let

$$\bar{\Phi}(q,p) = \begin{cases} \tilde{\Phi}_L(q,p) & \text{if } (q,p) \in \Gamma_L \\ (q,p) & \text{if } (q,p) \notin \Gamma_L \end{cases}$$

the application from Γ to Γ which corresponds to steps 3 and 4.

Remark 3. *For any point $(q,p) \in \Gamma$, if L is small enough, $(q,p) \in \Gamma_L$. However, if L is too large, Γ_L could be empty.*

2.2 Measure invariance via detailed balance

We recall the definition of detailed balance:

Definition 1 (detailed balance)**.** *A Markov chain P is said to satisfy detailed balance (or reversibility) with respect to the measure π if for every A and B measurable subsets,*

$$\int_B P(x,A)\pi(dx) = \int_A P(x,B)\pi(dx).$$

To prove detailed balance for P_π, we establish the following intermediate result:

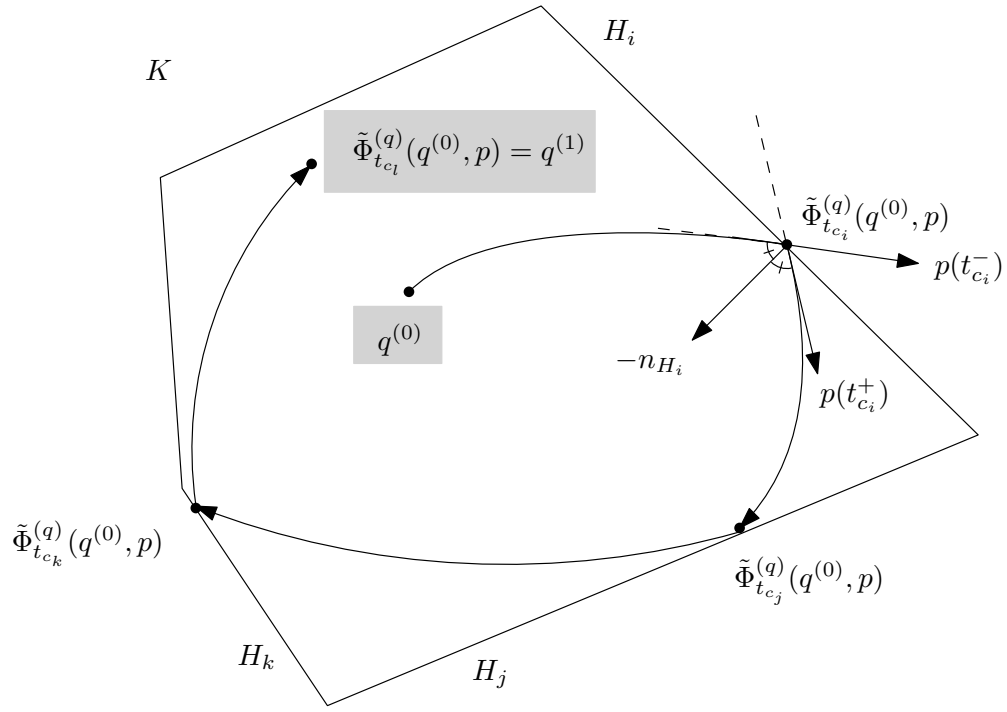


Figure 1: **One HMC step starting at $q^{(0)}$, with reflections of the HMC trajectory on the boundary of the polytope K.** The trajectory successively reflects on hyperplanes, before stopping at $q^{(n_L)}$. The normal to a facet is denoted n_H.

Theorem 1. *For a fixed time L, we consider the Markov kernel $P_{\pi,L}$ associated to steps 2 and 3 of Algorithm S1. Then $P_{\pi,L}$ satisfies detailed balance with respect to π.*

From which one derives the invariance of P_π:

Theorem 2. *The Markov kernel P_π associated to Algorithm S1 satisfies detailed balance with respect to π.*

It is well known that satisfying detailed balance implies the invariance of the measure.

2.3 Convergence result

Detailed balance ensures that the Markov kernel P_π leaves π invariant, but it does not imply the convergence of P_π^n to π by itself, with P_π^n is n-times iterated of P_π. In this section, we prove uniform ergodicity to get such a convergence result. This requires extra assumptions on Q.

Definition 2 is taken from [36] with P is a Markov kernel. Practically, we will use $P = P_\pi$.

Definition 2. *A subset $C \subset \mathcal{X}$ is small (or, (n_0, ϵ, ν)-small) if there exists a positive integer n_0, a real $\epsilon > 0$, and a probability measure $\nu(.)$ on $\mathcal{X}$ such that the following minorisation condition holds:*

$$P^{n_0}(x, .) \geq \epsilon\nu(.) \quad x \in C$$

i.e. $P^{n_0}(x, A) \geq \epsilon\nu(A)$ for all $x \in C$ and all measurable $A \subset \mathcal{X}$

Intuitively, the previous definition states that whatever the starting point x–whence the adjective small, the iterated kernels $P^{n_0}(x, .)$ cover a common measure $\nu(.)$. Note that the value of n_0 is the number of steps needed to reach ν – and can be equal to one. The following theorem provides a sufficient condition for Q to be small with respect to P_π:

Lemma 1. *If Q if convex and the gradient of the potential energy ∇U is bounded on Q, then Q is small for P_π.*

The ergodicity of P_π follows from the above Lemma 1, applied to Theorem 8 from [36]:

Theorem 3. *If Q if convex and the gradient of the potential energy ∇U is bounded on Q, then P_π is uniformly ergodic, that is, for all $x \in Q$:*

$$\|P_\pi^n(x, .) - \pi(.)\|_{TV} \leq (1 - \epsilon)^{\lfloor n/n0 \rfloor} \tag{3}$$

3 Application: computing the volume of a polytope

We now specialize Algorithm S1 and use it as a building block for polytope volume calculation from [33]. The general domain Q of Sect. 2 is now the polytope K.

3.1 Volume algorithm

In the following, we review the volume calculation algorithm, stressing the role of constants related to the random walk – as these play an important role when the dimension increases.

Overview. Consider a polytope defined in matrix form by $Ax \leq b$ with the origin 0 inside the polytope. The algorithm used in [33] computes the volume of such a polytope with target relative error ε. The principle is a multi-phase Monte Carlo computation, which splits the calculation into m steps. Let $\{f_0, \ldots, f_{m-1}\}$ be m isotropic Gaussian distributions i.e. $f_i(x) = \exp(-\|x\|^2/(2\sigma_i^2))$ with $\sigma_i = 1/\sqrt{2a_i}$, or equivalently

$$f_i(x) = \exp(-a_i\|x\|^2), \tag{4}$$

such that the first one is highly concentrated around a point deep inside the convex, and the last one is an almost flat distribution. The volume calculation reduces to computing the telescoping product

$$\mathrm{Vol}(K) = \int_K f_0(x)dx \frac{\int_K f_1(x)dx}{\int_K f_0(x)dx} \cdots \frac{\int_K dx}{\int_K f_{m-1}(x)dx} \equiv \int_K f_0(x)dx \prod_{i=1,\ldots,m} R_i \tag{5}$$

Each of these ratio R_i are estimated using Monte-Carlo integration with respect to the density

$$\pi_{i-1}(x) = \frac{f_{i-1}(x)}{\int_K f_{i-1}(y)dy} 1_K(x) \tag{6}$$

via importance sampling:

$$R_i = \frac{\int_K f_i(x)dx}{\int_K f_{i-1}(x)dx} = \int_K \frac{f_i(x)dx}{f_{i-1}(x)} \pi_{i-1}(x)dx. \tag{7}$$

The method then uses as core block an algorithm sampling the previous distribution, usually using HAR. In our case, HMC with reflections will be used instead.

Starting Gaussian $f_0(x)$. The constants involved in the choice of f_0 are not related to the random walk – *i.e.* this step does not use the random walk.

Cooling schedule incremental construction. The sequence of Gaussians $f_i(x)$ is chosen so as to estimate R_i efficiently. One takes $a_i = a_{i-1}(1-1/n)^r$, trying to maximize r so as to reach the uniform distribution as rapidly as possible. In practice, this is done by running a dichotomy on the values of r – one knows that $r = 1$ suffices, controlling the variance of the RV $Y = \exp((a_{i-1} - a_i))X)$. The following numbers of samples are used to estimate that the condition $\mathrm{Var}[Y]/\mathbb{E}[Y]^2 \leq C$ holds:

- HAR: $N_{cs} = 500 * C + n^2/2$ (From [33], with $C = 2$).
- HMC: $N_{cs} = 500 * C + n$ (Our algorithm, also with $C = 2$).

Cooling schedule termination. To decide whether the cooling schedule needs another Gaussian or the value m has been reached, the ratio R_i (Eq. 7) is evaluated with a number N_m of samples. If this ratio of close to one (≤ 1.001 in the `MATLAB` implementation of [33]), the cooling schedule is complete. In practice, this evaluation is carried out using $N_m = 150/\varepsilon$ samples returned by the random walk (`MATLAB` implementation of [33]).

Convergence of ratios R_i. For $X_1, ..., X_k$ consecutive points given by the random walk sampling π_{i-1}, the Monte-Carlo estimation $R_i^{(k)}$ of R_i is given by:

$$R_i^{(k)} = \frac{1}{k} \sum_{j=1}^{k} \frac{f_i(X_j)}{f_{i-1}(X_j)}. \tag{8}$$

The following stop criterion is introduced in [33]. Let $\varepsilon' = \varepsilon/\sqrt{m}$; this is the relative ratio error allocated for each ratio R_i estimation. Consider a sliding window of size W. When W consecutive estimated ratios $R_i^{(k-W+1)}, ..., R_i^{(k)}$ are within $\varepsilon'/2$, the convergence for R_i is declared.

In this work, we test the following window sizes:

- HAR with window size $W_{HAR} = 500 + 4 * n^2$ (From [33]).
- HMC0: HMC with window size $W_{HMC0} = 250$
- HMC1.5: HMC with window size $W_{HMC2} = 250 + n\sqrt{n}$

3.2 HMC algorithm

Our HMC specialization has analytical trajectories – see also [37], whose intersection with the polytope have simple expressions.

Analytical trajectories. We build an HMC sampler for $\pi_i(x)$ from Eq. 6. Let $U(q) = \log(\exp(-a_i||q||^2)) = -a_i||q||^2$. U can be interpret as a potential energy whose associated Boltzmann measure is π_i. The associated Hamiltonian in phase space is given by $H(q,p) = U(q) + 1/2\|p\|^2$. Note that the normalization constant $\frac{1}{\int_K f_i(y)dy}$ was discarded because it

does not change the trajectory (its gradient is 0). Rewriting the dynamical system associated to this Hamiltonian yields the following differential equations:

$$\frac{d^2 q_j}{dt^2}(t) = -2a_i q_j(t) \text{ for } j \leq n. \tag{9}$$

Each coordinate is independent and has a solution of the form

$$q_j(t) = C_j \cos(\omega t + \phi_j) \tag{10}$$

With $C_j, \omega, \phi_j \in \mathbb{R}$. The parameter Φ_j satisfies the following 2 equations:

$$\cos(\phi_j) = \frac{q_j(0)}{C_j}, \sin(\phi_j) = \frac{-p_j(0)}{\omega C_j}. \tag{11}$$

Thus we deduce:

$$\begin{cases} \omega & = \sqrt{2a_i} \\ C_j & = \sqrt{q_j(0)^2 + p_j(0)^2/\omega^2} \\ \phi_j & = \arctan\left(-\frac{p_j(0)}{q_j(0)\omega}\right) + 1_{\{q_j(0)<0\}}\pi \end{cases} \tag{12}$$

Note that these equations are the same for any choice of coordinates as long as the basis is orthonormal–an observation exploited below.

Collision with convex boundary. To restrict the sampling algorithm to the convex K, trajectories should reflect on the boundary of K. The convex K is defined by a set of hyperplanes. Hence, we need to compute the intersection time of a trajectory with each hyperplane and take the smallest time. Thankfully, there is an analytical expression. The hyperplanes are defined by a matrix A and a vector b, and hyperplane i is defined by the equation $(Ax)_i = b_i$.

To compute the collision time with a hyperplane H, we make the following remark: let n_H be the normal of the hyperplane. We can complete n_H to an orthonormal basis. In this basis, the collision time depends only on what happens for the coordinate on n_H. Consider $q_{n_H}(t) =< q(t), n_h >$ and $p_{n_H}(t) =< p(t), n_H >$ the coordinates along the normal of the hyperplane. Let ω,C_{n_H} and ϕ_{n_H} be the parameters of the trajectory along direction n_H. Finding the intersection times is equivalent to solving the equation for t:

$$C_{n_H} \cos(\omega t + \phi_{n_H}) = b_i \tag{13}$$

We deduce that if $|C_{n_H}| < b_i$, there is no solution, and else, the following times are solution:

$$\begin{cases} t_1 & = \left(\arccos(b_i/C_{n_H}) - \phi_{n_H}\right)/\omega. \\ t_2 & = \left(-\arccos(b_i/C_{n_H}) - \phi_{n_H}\right)/\omega. \end{cases} \tag{14}$$

One solution corresponds to the entry into K, while the other corresponds to the exit out of K. We select the exit trajectory via a dot product between the velocity at t_1 and t_2 and the outward normal n_H. In the sequel, the corresponding value is denoted t_c for time of collision.

3.3 Travel time choice with respect to a_i

Algorithm S1 can be slightly generalized by choosing a travel time L uniformly in $[0, L_{max}]$ instead of $[0, 1]$. We propose here a strategy to chose L_{max} with respect to the parameter a_i of the Gaussian sampled ($\sigma_i = 1/\sqrt{2a_i}$).

We distinguish two main cases here. First, if a_i is large, then the probability measure is concentrated in a neighborhood of 0, and the trajectories will seldom hit the boundaries and the strategy is largely unaffected by the convex. Second, if a_i is close to 0, the distribution is close the uniform distribution of the convex.

Case 1. If a_i is large, we consider $q_j(t)$ a solution of the dynamical system 9 and we introduce space-time rescaling for the trajectory:

$$\bar{q}_j(t) = \sqrt{a_i} q_j(t/\sqrt{a_i}).$$

It satisfies Eq. 9 with $a_i = 1$:

$$\frac{d^2\bar{q}_j}{dt^2}(t) = \bar{q}_j(t)$$

with initial condition

$$\frac{d\bar{q}_j}{dt}(0) = \frac{dq_j}{dt}(0) \sim \mathcal{N}(0, 1). \tag{15}$$

It follows that in the absence of convex boundaries, the rescaled process $\bar{q}$ is sampling the distribution associated to $a_i = 1$, i.e. the standard normal distribution. Therefore, any sensible value for $L_{max}(1)$ taken for $a_i = 1$ should be scaled as $L_{max}(a_i) = L_{max}(1)/\sqrt{a_i}$.

Case 2. When a_i goes to 0, the previous strategy leads $L^{a_i}_{max} \to \infty$. We argue that in this case, the trajectories converges to billiard trajectories with reflections on the boundary and that the optimal L_{max} should be therefore close to the optimal L^0_{max} for $a_i = 0$. To the best of our knowledge, this optimal value is not known. However as we shall see in section 3.5, if the polytope is a cube, choosing L^0_{max} such that the number of reflection on the boundary is proportional to the dimension leads to a $O(log(n))$ mixing time (where n is the dimension). Hence we propose a heuristic where for a given parameter r, L^0_{max} is chosen such that on average for $a_i = 0$ the trajectory hits the boundary $r \times n$ times. Practically, we perform a binary search on the values of L^0_{max}, and for each candidate value evaluate the average number of reflections over $N_0 (= 1000)$ steps.

Values used in experiments. Mixing the two cases, we chose L_{max} as follows:

$$L_{max}(a_i) = \min(\pi/\sqrt{2a_i}, L^0_{max}) \tag{16}$$

3.4 HMC implementation based on interval arithmetic

3.4.1 Robustness issues

The algorithm as stated before can be implemented in the real RAM model (Algo. S1), but is prone to numerical rounding errors. As a particular case, one may consider the situation

where rounding errors would be such that the point computed on the HMC trajectory would be outside the convex. More generally, all geometric constructions and geometric predicates on them potentially raise robustness issues – see list in section 3.4.3.

As discussed in Introduction, we guarantee robustness using the SEGC paradigm. More specifically, recall that efficient arithmetic operations usually combine two ingredients: first, an interval representation of the numbers, as non overlapping intervals yield exact predicates; second, an arbitrary precision representation of the interval bounds, as precision can be increased so as to yield exact predicates and constructions of controlled accuracy. In the sequel, we use the `iRRAM` library which provides these two ingredients [32].

For the sake of clarity, all functions for which the `iRRAM` library plays a key role are highlighted in blue.

3.4.2 `iRRAM` and used features

We represent points as n-dimensional points whose coordinates are of the `iRRAM` number type. In `iRRAM`, a real number is represented by two types of data: firstly a symbolic representation memorizing the way it was defined (type of function and pointers to the operands), and secondly a numeric approximation using an interval with rational endpoints guaranteed to enclose the exact real value. The accuracy of the latter interval can be increased if needed, by recomputing it using recursively increased precision of the operands intervals that defines it. Therefore, the `iRRAM` encoding $q^{\text{iRRAM}}(t)$ of the position $q(t)$ of the HMC trajectory is numerically represented by a n-dimensional box certified to contain the exact real position.

Two specific operations of `iRRAM` may trigger numerical refinement:

- operator $x < y$: predicate answering the $<$ comparison operator. If the interval representations of x and y overlap, these intervals are automatically refined, a feature called *precision refinement* thereafter.
- Near_inf_double(iRRAM x) : returns the nearest double $<$ the `iRRAM` number x.

3.4.3 Robust operations and robust HMC step

Our robust implementation calls for the following operations (i) Computing the trajectory parameters – Eq. 12, (ii) Finding the exit intersection time – Eq. 14, (iii) Finding the smallest exit intersection time amongst all hyperplanes, and (iv) Evolving the trajectory. In the sequel, we detail these operations, and refer the reader to the SI section 7; in particular, Algorithm S2 refines Algorithm S1 based on these robust primitives.

Trajectory parameters. Following Eq. 12, we construct the numbers C_j, w and ϕ_j as `iRRAM` number types. See Algorithm S4.

Exit intersection time t_c with one hyperplane. Intersecting the trajectory with a hyperplane yields two solutions (Eq. 14) respectively exiting and entering the convex. The collision time t_c corresponds to the former. Assuming that the normal vector to the

hyperplane is oriented outwards, the value t_c is such that $< p(t_c), n_H >> 0$. The evaluation of this predicate triggers a precision refinement if needed.

Smallest exit intersection time. Since the boundary of K involves several hyperplanes, the nearest one, which corresponds to the smallest exit time, must be determined. To do so, we first construct the intersection time t_{c_i} with respect to each hyperplane. Then we compute $t_c = \min_i t_{c_i}$.

This calculation is tantamount to sorting the individual intersection times, which in turn requires the comparison operator $<$. `iRRAM` provides such an operator, which triggers precision refinement if needed. See Algorithm S5.

Remark 4. *We note that in case the trajectory would hit a face of dimension $< d-1$, an equality between exit times occurs. The absence of separation bound in* `iRRAM` *does not allow us to handle such cases, and an infinite refinement loop is entered. However, for each starting point, the measure of velocities leading to such sets is null. Practically, such a case was never faced–as expected.*

The Is_strictly_in_convex$[NT](Point\ q)$ predicate. To constrain the trajectory within the convex, we resort to a predicate using a number type NT, telling whether a given position q belongs to the interior K^{o} of K. The predicate checks whether $< op, n > < b_i$ holds for every hyperplane. (Nb: $< \cdot, \cdot >$ stands for the dot product of two vectors.)
The robust implementation Is_strictly_in_convex[iRRAM](iRRAM_point_d p) of this predicate uses `iRRAM` as number type, triggering the `iRRAM` refinement if needed.

Robust implementation of one HMC step. The robust implementation of one step hinges on two operations (Algorithm S2):

- Computing the nearest inferior double $t_c^< =$ Near_inf_double(t_c). The intersection point between the trajectory and a hyperplane is defined analytically by Eq. 14. The `iRRAM` representation of the collision time is an interval certified to contain the exact solution, and the corresponding n-dimensional point $q^{\mathrm{iRRAM}}(t_c^<)$ is represented as a box. We note that the box $q^{\mathrm{iRRAM}}(t_c^<)$ intersects the interior K^{o} of the convex K. Indeed:
  - the exact collision point $q(t_c)$ lies on its defining hyperplane i.e. $q(t_c) \in H_i$
  - by definition of $t_c^<$, the exact embedding $q(t_c^<)$ satisfies $q(t_c^<) \in K^{\mathrm{o}}$
  - the `iRRAM` box $q^{\mathrm{iRRAM}}(t_c^<)$ corresponding to $t_c^<$ intersects K^{o} since

$$q^{\mathrm{iRRAM}}(t_c^<) \ni q(t_c^<) \in K^{\mathrm{o}} \tag{17}$$

- Calling the predicate Is_strictly_in_convex[iRRAM]$(q(t_c^<))$. Recall that in `iRRAM`, a n-dimensional point is represented as a box. This is in particular the case for the collision points with the hyperplanes, and for the final point returned. To ensure that all such points are strictly within the convex, we call the aforementioned Is_strictly_in_convex[iRRAM].

3.4.4 Mixed precision and MATLAB interface

In practice, a robust implementation faces two issues: it is much slower than the corresponding double precision implementation, and it currently has to be interfaced with MATLAB, which cannot handle iRRAM number types.

To solve the first problem, we use a lazy two step strategy (Algorithm S3). First, the HMC step is computed using double precision. Then the programs checks whether the resulting point is inside the convex, using the CGAL library. If not, the computation is restarted using the robust iRRAM implementation.

The second issue forces us to convert the resulting point from a representation using the iRRAM number type (when relevant) to a representation using doubles. This conversion may result in a point outside of the convex. We check this property using the Is_strictly_in_convex predicate, using here a lazy approach again:
first, Is_strictly_in_convex[CGAL::Interval_nt_advanced] is computed; in case of failure, Is_strictly_in_convex[CGAL::Lazy_exact_nt<CGAL::Quotient<CGAL::MP_Float> >] is evaluated [38]. When the point is detected outside the convex, we refuse it, resulting in a move step keeping the initial point invariant.

Remark 5. *Having started strictly inside the convex, with respect to the SEGC paradigm (Sec. 1.3), we never faced a situation* I can't compute.

3.5 Cube: HMC mixing time is $O(\log n)$

We make a small digression in the case of the cube. It turns out that in this special case, the mixing time of the HMC random walk is $O(\log n)$ with n the dimension, and the average complexity per step (the average number of calls to the oracle per step) is linear with the dimension. We assume without any loss of generality that the cube is $[0,1]^n$. Rigorously:

Theorem 4. *Let $P_k^{(n)}$ the distribution after k steps in dimension n and g_n an isotropic Gaussian of parameter a_i restricted to $[0,1]^n$. Then there exists $0 < \rho < 1$ such that for every $\epsilon > 0$, every $n > 1$ and every $x \in \mathbb{R}^n$, we have for $k \geq (\log n - \log \epsilon)/(\log 1/\rho)$:*

$$\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq \epsilon. \tag{18}$$

Furthermore, the average number of reflections per step is $O(n)$.

4 Experiments

4.1 Implementation and code availability

Implementation. The implementation of our algorithm for the particular case of a convex polytope $Q = K$ is provided in the Structural Bioinformatics Library (SBL, http://sbl.inria.fr), a state-of-the-art environment targeting molecular simulation at large

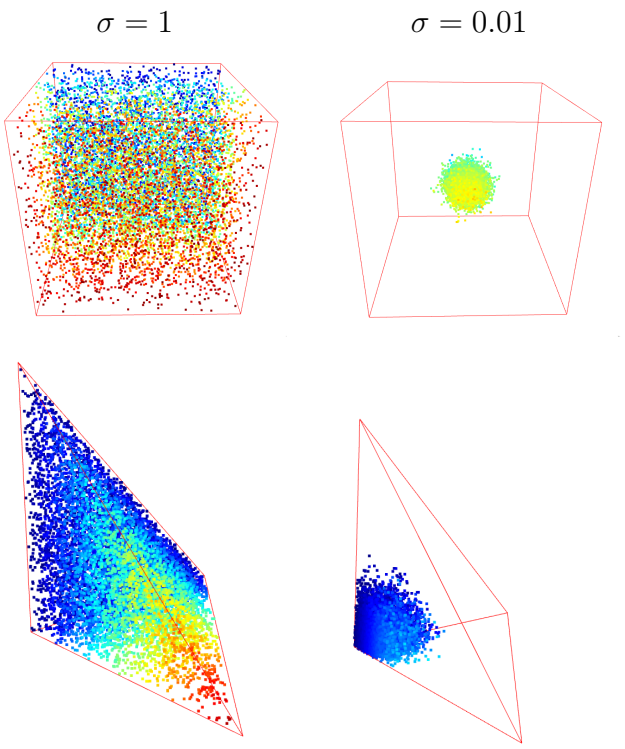


Figure 2: **Using HMC to sample a Gaussian of standard deviation σ restricted to the cube $[-1, 1]^3$ and the standard simplex.** In all cases, $N = 10,000$ samples. (Color gradient defined w.r.t. a coordinate value.)

[39]. The corresponding package, Hamiltonian_Monte_Carlo is ascribed to the *Core / Geometry-Topology* component of the library. The user manual of the package, as well as a jupyter notebook illustrating the main functionalities, can be accessed from `https://sbl.inria.fr/doc/Hamiltonian_Monte_Carlo-user-manual.html`.

Volume calculations. As described in section 3, we also embed our random walk in the framework of [33]. We reuse the `MATLAB` code provided by [33] and adapt it so as to call our HMC random walk instead of the usual HAR random walk. Also following [33], we use HAR (rather than ball walk) as contender.

4.2 Illustrations of the HMC random walk

Our first illustration features samples generated by HMC for the cube $[-1, 1]^3$ and the standard simplex $\sum x_i \leq 1$ in dimension three. Varying the parameter σ rapidly yields concentrated samples (Fig. 2). Our second illustration shows the ability of the algorithm to escape corners. As opposed to HAR, HMC yields an almost uniform distribution after 10 steps even when the dimension increases (Fig. 3).

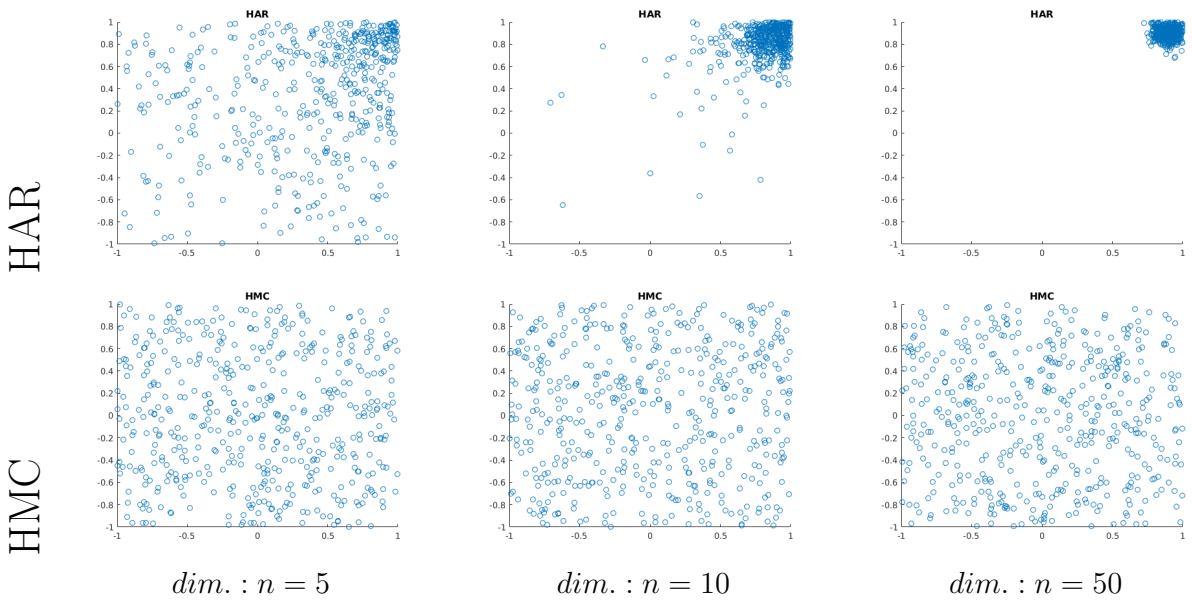


Figure 3: **HMC for the cube $[-1,1]^n$: ability to escape corners, and comparison to Hit-and-run.** A nearly flat isotropic Gaussian distribution was used to approach the uniform distribution ($\sigma = \sqrt{500}$). For $n > 3$, the plot displays projections onto the first two coordinates. Simulations were started from a corner ($q_i^{(0)} = 0.9$). Samples generated after 10 steps of HAR or HMC, repeated 500 times – whence 500 samples.

4.3 Experimental setup

Models. In [33], five types of polytopes are used for testing: Birhkoff polytopes, triply-stochastic polytopes, the cube, the isotropic simplex, and the standard simplex. To present detailed statistics on the quality of the output, we focus on the latter three, since their exact volume can be computed. We perform a systematic study in dimension $n = 10, 30, 50, 70, 100$. As target error, we use $\varepsilon = 0.01$ – see Section 3.1. We also present selected results in dimension $n = 250$ and $n = 500$, with $\varepsilon = 0.05$.

Algorithms and parameters used. As specified in Section 3.1, volume algorithms based on HAR and HMC require three sets of constants: N_{cs} for the cooling schedule incremental construction, N_m for the cooling schedule termination, and the window size W to assess the convergence of ratios R_i. In addition, fixing the travel time in HMC requires the parameter r (see the discussion accompanying Eq. 16).

As specified in Section 3.1, we compare three contenders HAR, HMC0, and HMC1.5.Table 1 summarizes all parameters.

To perform statistics, 50 runs of each algorithm on a given polytope of prescribed dimension were obtained. Summarizing, our experiments involve 3 algorithms $\times$ 3 polytopes $\times$ 5 dimensions $\times$ 50 repeats – whence a total of 2250 runs.

	N_{cs}	N_m	W	r
HAR	$1000 + n^2/2$	$150/\varepsilon$	$500 + 4 * n^2$	NA
HMC0	$1000 + n$	$150/\varepsilon$	250	0.1
HMC1.5	$1000 + n$	$150/\varepsilon$	$250 + n\sqrt{n}$	0.1

Table 1: **Summary of parameters used for all contenders.** N_{cs}: number of samples used for the incremental construction of the cooling schedule; N_m: number of samples used in the cooling schedule termination condition; W: window size to estimate the R_is; r: parameter used to adjust the travel time. Nb: ε is the target quality.

Remark 6. *Our implementation do not handle the degenerated case $a_{flat} = 0$, hence we use $a_{flat} = 10^{-13}$ (Eq. 4) to tune the travel time for the flat Gaussian.*

Statistics. We collect the following statistics:

- the relative error of the estimated $\tilde{V}$:

$$\text{err}_r = \mid \tilde{V} - \text{Vol}(K) \mid / \text{Vol}(K). \tag{19}$$

- $\#S$: number of sampled points for a volume computation.
- $\#O$: complexity, i.e. the number of calls to the oracle. For HAR, this is equal to the number of sampled points. For HMC, $\#O$ takes the number of reflections into account.

- $\#R$: multi-precision refinements triggered in `iRRAM` (Algo. S3). Nb: `MATLAB` first attempts a calculation using double floating point numbers. In case of failure, a precision refinement is triggered.

- $\#E$: the number of *exits* of the polytope (Algo. S3). As discussed in Section 3.4.4, this may happen when converting the coordinates of a point from the `iRRAM` representation to the double representation. Note that such events are not related to our implementation, but instead to the number type (doubles) used in the `MATLAB` implementation of the volume algorithm.

- t: the running time, t in seconds.

Computer. Calculations were run on a desktop DELL Precision 7920 Tower (Intel Xeon Silver 4214 CPU at 2.20GHz, 64 Go of RAM), under Linux Fedora core 32.

4.4 Results

We first analyze results up to dimension $n = 100$.

Quality of estimates vs running time. Analyzing the relative error err_r and the running time t per se, as expected, a clear correlation between the median values and the stdev appears (Fig. S1, Table S1). In terms of comparisons, the global trend states that HMC0 is faster than HMC1.5, which is faster than HAR, a fact especially visible for large values of the dimension, whatever the polytope.

Next, we use scatter plots to relate the relative error (median, stdev) to the running time.

The first two scatter plots link symbols for a given polytope/dimension, which is instrumental to compare the contenders (Fig. 4(Tow row)). We first notice that compared to the simplices, the cube appears as an easy case. For the two simplices, HMC is faster and more accurate than HAR; also, HMC1.5 is more accurate but slower than HMC0. As a specific illustration, consider the standard simplex in dimension $n = 100$: HMC0 is ~ 16 times faster than HAR, achieving a median accuracy 7.34% against14.9% for HAR. HMC1.5 gets an accuracy of 4.7% while remaining ~ 5.7 times faster than HAR.

The second two scatter plots link symbols for a given algorithm, which is useful to assess the incidence of dimension (Fig. 4(Bottom row)). For HMC, the median and the stdev errors remain in a *narrow band*, that is, these order one and two moments do not increase as a function of the dimension. We also note that statistics for HMC1.5 are more stable than those of HMC0. For HAR, one needs to distinguish cases. As already noticed, the cube is an easy case, for which a behavior similar to HMC is observed. For the two simplices, both the median and the stdev of the error significantly increases beyond the dimension $n = 50$.

From these observations, depending on the desired trade-off between accuracy an speed, one may choose HMC0 or HMC1.5.

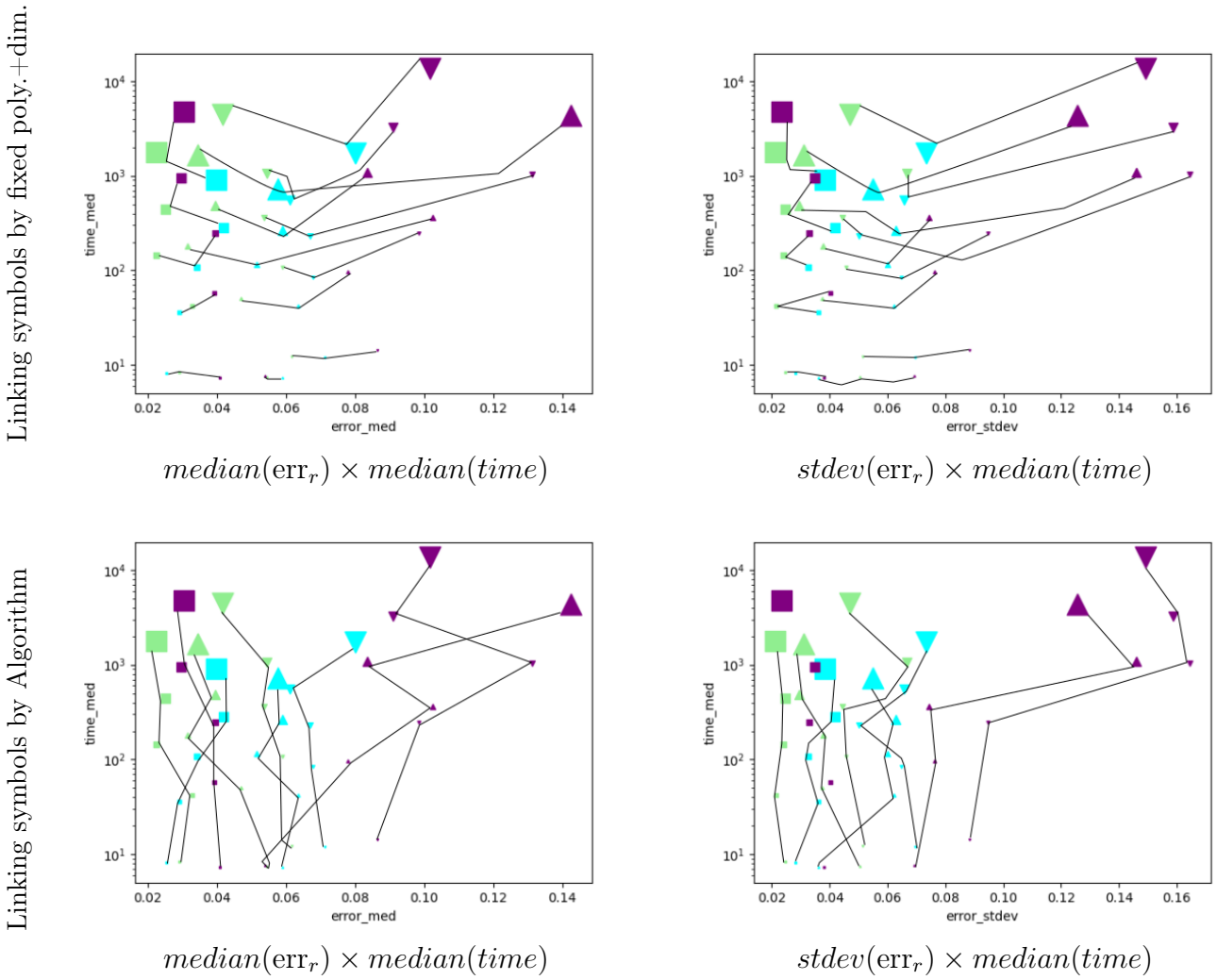


Figure 4: **Estimating the volume of three polytopes with known volumes (cube, isotropic simplex, standard simplex), in dimension $n = 10, 30, 50, 70, 100$, with three algorithms.** Statistics reported over 50 runs. The error of an estimate $\tilde{V}$ is defined as $\mathrm{err}_r = |\tilde{V} - V|/V$. **Symbol shape vs polytope:** square: cube; triangle up: isotropic simplex; triangle down: standard simplex **Symbol size vs dimension :** smallest for $n = 10$, largest for $n = 100$ **Color vs algorithm:** purple/light blue/light green HAR/HMC0/HMC1.5 **(Top row)** Comparing the three algorithms on a given polytope (type and dimension fixed). **(Bottom row)** Assessing the incidence of dimension by running a given algorithm on instances of a fixed polytope type whose dimension decreases.

Other statistics - robustness. The main issue with trajectories computed with doubles is that they may exit the polytope and remain outside: this is observed in practice, with selected runs using double floating point numbers never ending. Our robust implementation fixes this difficulty. As a quantitative measure, we track the number of times $\#R$ precision

refinement is triggered in iRRAM– see Section 4.3. It clearly appears that multiprecision gets used more often when the dimension increases, say in the range 50-100 for our experiments (Table S2). This is particularly striking for the standard simplex, with a number of calls in thousands in dimension $n = 100$. We speculate that this is due to local geometry properties near corners of the simplex. For HMC, another interesting statistic is $\#O/\#S$, namely the number of reflections per step. As expected, the ratio between the number of reflections and the number of step is a bit smaller than the target $r \times n$ (Section 3.3), since for most of the Gaussian the length of the trajectory will be smaller than L^0_{max}.

Results in dimension $n = 250, 500$. We now briefly comment on results for higher dimensional cases (Tables S3, S4, S5, S6).

For the dimensions tested, only the cube is handled by the implementation of the volume algorithm, due to the format of the floating point numbers used in MATLAB. (Note that the exact volume is not computed either due to this issue.) Using $\varepsilon = 0.05$, we notice that HMC manages to estimate the order of magnitude of the volume of the cube satisfactorily. As for robustness, we note that precision refinements are called often, especially for the standard simplex, an observation already raised for $n = 100$.

5 Conclusion

To compute the volume of high dimensional polytopes, this paper exploits a novel strategy based on billiard Hamiltonian Monte Carlo. This strategy exploits the ability of billiard walks to escape corners, and the simple analytical expression of the Hamiltonian makes a robust but fallible implementation based on our SEGC paradigm possible and effective, as shown by experiments on polytope volume calculations using the iRRAM and CGAL libraries.

On the theoretical side, our work leaves stimulating questions open, two of which are of prominent importance. The first one is the choice of the optimal travel time required to sample a convex uniformly, which would ideally be determined at runtime for each convex. The second one is the analysis of the incidence of reflections on the mixing time, so as to quantify the speed at which reflections decorrelate successive points.

Our work also prompts a connection with statistical physics, for the calculation of density of states (DoS), which measure the volume in phase space of the pre-image of an energy stratum. DoS are central to compute partition functions, whence all thermodynamic quantities. We anticipate that billiard on level set surfaces bounding strata will prove beneficial, with a potential impact in statistical physics at large.

Acknowledgments. We are grateful to B. Cousins and S. Vempala [33] for providing a high quality MATLAB code.

References

[1] M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988.

[2] I. Bárány and Z. Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987.

[3] S. Levy. *Flavors of Geometry*. Cambridge University Press, 1997.

[4] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38(1):1–17, 1991.

[5] László L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006.

[6] B. Cousins and S. Vempala. Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In *ACM STOC*, pages 539–548. ACM, 2015.

[7] Yin Tat Lee and Santosh S Vempala. Convergence rate of riemannian hamiltonian Monte Carlo and faster polytope volume computation. In *STOC*, pages 1115–1121. ACM, 2018.

[8] B. Cousins and S. Vempala. Gaussian cooling and $O^*(n^3)$ algorithms for volume and gaussian volume. *SIAM Journal on Computing*, 47(3):1237–1273, 2018.

[9] Apostolos Chalkis, Ioannis Z Emiris, and Vissarion Fisikopoulos. Practical volume estimation by a new annealing schedule for cooling convex bodies. *arXiv preprint arXiv:1905.05494*, 2019.

[10] S. Brooks, A. Gelman, G. Jones, and X-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[11] D. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[12] L. Lovász. Hit-and-run mixes fast. *Mathematical Programming, Series B*, 86:443–461, 12 1999.

[13] L. Lovász and S. Vempala. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

[14] L. Lovász and S. Vempala. Hit-and-run from a corner. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 310–314, New York, NY, USA, 2004. ACM.

[15] H. Haraldsdóttir, B. Cousins, I. Thiele, R. Fleming, and S. Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 2017.

[16] L. Lovász and R. Kannan. Faster mixing via average conductance. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 282–287, New York, NY, USA, 1999. ACM.

[17] I. Emiris and V. Fisikopoulos. Efficient random-walk methods for approximating polytope volume. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 318. ACM, 2014.

[18] I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. on Math. Software*, 44(4), 2018.

[19] Y. T. Lee and S. S. Vempala. Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 998–1007, 2017.

[20] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[21] N. Radford. MCMC using Hamiltonian Dynamics. In Galin L. Jones Steve Brooks, Andrew Gelman and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, chapter 5, pages 113–162. Chapman & Hall/CRC, 2012.

[22] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[23] Felipe Alvarez, Jérôme Bolte, and Olivier Brahic. Hessian riemannian gradient flows in convex programming. *SIAM journal on control and optimization*, 43(2):477–501, 2004.

[24] Margaret H Wright. Some properties of the hessian of the logarithmic barrier function. *Mathematical Programming*, 67(1-3):265–295, 1994.

[25] Howard Masur and Serge Tabachnikov. *(Chapter 13) Rational billiards and flat structures*, pages 1015–1089. Number PART A in Handbook of Dynamical Systems. Part a edition, December 2002.

[26] Péter Bálint, Nikolai Chernov, Domokos Szász, and Imre Péter Tóth. Geometry of multi-dimensional dispersing billiards. In Wellington de Melo, Marcelo Viana, and Jean-Christophe Yoccoz, editors, *Geometric methods in dynamics (I) : Volume in honor of Jacob Palis*, number 286 in Astérisque, pages 119–150. Société mathématique de France, 2003.

[27] B. Polyak and E.N. Gryazina. Billiard walk-a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes*, 47(3):6123–6128, 2014.

[28] H. Afshar and J. Domke. Reflection, refraction, and Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3007–3015, 2015.

[29] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

[30] C. Yap and T. Dubé. The exact computation paradigm. In *Computing in Euclidean Geometry*, pages 452–492. World Scientific, 1995.

[31] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[32] N. Müller. The iRRAM: Exact arithmetic in C++. In *Computability and Complexity in Analysis*, pages 222–252. Springer, 2001.

[33] B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.

[34] E. Gryazinaa and B. Polyak. Random sampling: Billiard walk algorithm. *arXiv*, 2014.

[35] Nikolai Chernov and Roberto Markarian. *Chaotic billiards.* Number 127. American Mathematical Soc., 2006.

[36] G.O. Roberts and J.S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 2004.

[37] A. Pakman and L. Paninski. Exact hamiltonian Monte Carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

[38] A. Fabri and S. Pion. A generic lazy evaluation scheme for exact geometric computations. In *Proc. 2nd Library-Centric Software Design*, pages 75–84, 2006.

[39] F. Cazals and T. Dreyfus. The Structural Bioinformatics Library: modeling in biomolecular science and beyond. *Bioinformatics*, 7(33):1–8, 2017.

[40] James V. Burke. Continuity and differentiability of solutions. `https://sites.math.washington.edu/~burke/crs/555/555_notes/continuity.pdf`.

6 Proofs

6.1 Lemmas for Thm. 2

Let A and B be measurable subsets of Q. Then let

$$A_B = \{(q,p) \in \Gamma_L | q \in A, \bar{\Phi}(q,p) \in B \times \mathbb{R}^n\}$$

be the subset of Γ of all positions in A with momenta that brings them in B after time L. Similarly, we define on Γ

$$\Psi(q,p) = (\bar{\Phi}^{(q)}(q,p), -\bar{\Phi}^{(p)}(q,p)). \tag{20}$$

Lemma 2. *The maps $\bar{\Phi}$ and Ψ preserve the Lebesgue measure on Γ_L. ie for every $A \subset \Gamma_L$ measurable, $\lambda(\bar{\Phi}^{-1}(A)) = \lambda(A)$*

Lemma 2. Clearly it is enough to prove that $\tilde{\Phi}_t$ preserves the Lebesgue measure. In [28], it is proved that for a fixed step size, the Euler method for numerical integration applied to the Hamiltonian flow with reflections, gives rise to a flow that preserves the Lebesgue measure. Letting the step size going to zero, we see that $\tilde{\Phi}_t$ is the pointwise limit of transformations that preserves the Lebesgue measure which implies that $\tilde{\Phi}_t$ preserve the Lebesgue measure. □

Lemma 3. *1. $\Psi(\Gamma_L) \subset \Gamma_L$*

2. $\Psi \circ \Psi = I$ on Γ_L

3. For any measurable sets A and B of $\mathbb{R}^n$,

$$B_A = \Psi(A_B)$$

4. $H(\Psi(q,p)) = H(q,p)$ for all $(q,p) \in \Gamma$.

Lemma 3. 1. and 2. are simple consequences of the reversibility of the Hamiltonian flow with reflections.
3. Let A and B be measurable sets of $\mathbb{R}^n$.
$\Psi_q(A_B) \subset B$ and $\Psi(\Psi(A_B)) = A_B$ thus $\Psi(A_B) \subset B_A$.
By symmetry, $\Psi(B_A) \subset A_B$. Hence by composing with Ψ: $\Psi(\Psi(B_A)) \subset \Psi(A_B)$. Using 2., we deduce $B_A \subset \Psi(A_B)$.
4. is clear.

□

Thm. 1. Let A and B be measurable sets of $\mathbb{R}^n$. The left hand side of the detailed balance equation becomes

$$\begin{aligned}
\int_A P_{\pi,L}(q,B)d\pi(q) &= \int_A P_{\pi,L}(q,B)\exp(-U(q))dq \\
&= \int_A \Big[\int_{\{p|(q,p)\in\Gamma_L, \bar{\Phi}^{(q)}(q,p)\in B\}} \exp(-\|p\|^2)dp \\
&+ \int_{\{p|(q,p)\notin\Gamma_L, \bar{\Phi}^{(q)}(q,p)\in B\}} \exp(-\|p\|^2)dp \Big] \exp(-U(q))dq \\
&= \int_{A_B} \exp(-H(q,p))dqdp + \int_A \int_{\{p|(q,p)\notin\Gamma_L\}} 1_B(q)\exp(-H(q,p))dqdp \\
&= \int_{A_B} \exp(-H(q,p))dqdp + \int_{A\cap B} \int_{\{p|(q,p)\notin\Gamma_L\}} \exp(-H(q,p))dqdp.
\end{aligned}$$

By symmetry:

$$\int_B P_{\pi,L}(q,A)d\pi(q) = \int_{B_A} \exp(-H(q,p))dqdp + \int_{A\cap B} \int_{\{p|(q,p)\notin\Gamma_L\}} \exp(-H(q,p))dqdp$$

Using lemma 3 and the measure conservation of lemma 2 we obtain:

$$\begin{aligned}
\int_{A_B} \exp(-H(q,p))dqdp &= \int_{\Psi(B_A)} \exp(-H(q,p))dqdp \\
&= \int_{B_A} \exp(-H(\Psi(q,p)))dqdp \\
&= \int_{B_A} \exp(-H(q,p))dqdp
\end{aligned}$$

Which concludes the proof. □

6.2 Proof of Lemma 1

Lemma 4. *Let Q be an open convex subset in $\mathbb{R}^n$ that contains the ball $B(0,2\rho)$, let x be a point in Q and let v be in $B(0,\rho)-x$. Let $f:\mathbb{R}^n\to\mathbb{R}^n$ be a continuous and bounded map. Suppose that $(x(t),v(t))\in\mathbb{R}^{2n}$ is the solution of the Cauchy problem*

$$\begin{aligned}
x(0) &= x \\
v(0) &= v \\
x'(t) &= v(t) \\
v'(t) &= af(t)
\end{aligned}$$

where a is a real number. If $|a|\leq\frac{\rho}{\|f\|_\infty}$ then for all $t\in[0,1]$, $x(t)$ is in the convex hull of x and of the ball $B(0,2\rho)$ and therefore in Q.

Lemma. 4. Suppose $|a| \leq \frac{\rho}{\|f\|_\infty}$. By the mean value theorem, $v(t) = v + w(t)$ where $\|w(t)\| \leq |a|\|f\|_\infty t \leq \rho t$ for $t \geq 0$. The derivative of function $y(t) = x(t) - vt$ is $w(t)$, therefore by the mean value theorem, $y(t) = y(0) + tz(t)$ where $\|z(t)\| \leq \rho t/2$. Therefore for all $t \in [0, 1]$,

$$\begin{aligned} x(t) &= y(t) + vt \\ &= (1-t)x + t((v+x) + z(t)) \end{aligned}$$

is in the convex hull of x and of the ball $B(0, 2\rho)$. □

Lemma 5. *Let B be an open ball in $\mathbb{R}^n$ and let $\varphi : B \to \mathbb{R}^n$ be a differentiable map. If for each x in B, $\|d\varphi(x) - Id\| < 1$, then φ is a diffeomorphism.*

Lemma 5. The only thing to prove is that φ is one to one. Let $x \neq y$ be two points in B. Consider the map $f : t \in [0, \|y - x\|] \to \varphi(x + tu) \cdot u$ where $u = \frac{y-x}{\|y-x\|}$. Using Schwarz inequality and the assumption we obtain

$$\begin{aligned} f'(t) &= d\varphi(x + tu)(u) \cdot u \\ &= Id(u) \cdot u + (d\varphi(x + tu) - Id)(u) \cdot u \\ &\geq 1 - \|d\varphi(x + tu) - Id\|\|u\|\|u\| \\ &> 0. \end{aligned}$$

It follows that $f(\|y - x\|) > f(0)$. Now $f(0) = \varphi(x) \cdot u$ and $f(\|y - x\|) = \varphi(y) \cdot u$, hence $\varphi(x) \neq \varphi(y)$. □

Lemma 6. *Let $B = B(0, r)$ be a closed ball in $\mathbb{R}^n$ of center 0 and radius $r > 0$ and let $\varphi : B \to \mathbb{R}^n$ be a one to one continuous map. If for all x in B, $d(x, \varphi(x)) \leq r/4$ then $\varphi(B)$ contains the ball $B(0, r/2)$.*

Lemma 6. By Jordan-Brouwer Theorem, the complement in $\mathbb{R}^n$ of the image Σ of the sphere $S = \partial B$ has exactly two connected components C_1 and C_2, one which is bounded, say C_1, and one which is not. By assumption the open ball $\overset{\circ}{B}(O, r/2)$ doesn't intersect Σ, hence is included in C_1 or C_2.

The image $E = \varphi(\overset{\circ}{B})$ of the interior of the ball B is included in C_i, one of the two connected components of $\mathbb{R}^n \setminus \Sigma$. On the one hand, by Jordan-Brouwer invariance of the domain theorem, E is open. On the other hand, $E = \varphi(B) \cap C_i$ and since $\varphi(B)$ is compact, $C_i \setminus E$ is open. Now C_i is connected, hence E or $C_i \setminus E$ is empty. Therefore $E = C_i$.

Since $\varphi(B)$ is compact, E is bounded, and therefore $E = C_i = C_1$. Moreover, by assumption, $\varphi(0) \in B^o(0, r/2) \cap E$ which implies that $B^o(0, r/2) \subset E$. □

Lemma 1. We assume without any loss of generality that $0 \in Q$.

Let $q^{(1)}, q^{(2)} \in Q$. One way to get a trajectory going from the point $q^{(1)}$ to a point very close to $q^{(2)}$, is to select a very high momentum $p_\alpha = \frac{1}{\alpha}(q^{(2)} - q^{(1)})$ and a short time $t_\alpha = \alpha$ with $\alpha > 0$. When $\alpha \to 0$, the potential energy term U of the Hamiltonian becomes less and less relevant, thus the trajectory converges to a straight line and $\lim_{\alpha \to 0} \Phi^q_{t_\alpha}(q^{(1)}, p_\alpha) = q^{(2)}$.

This intuition is formalized using the scaled variables

$$\begin{cases} \tilde{q}(t) & = q(\alpha t) \\ \tilde{p}(t) & = \alpha p(\alpha t). \end{cases} \tag{21}$$

The equation of motion becomes:

$$\frac{d\tilde{q}}{dt}(t) = \tilde{p}(t) \tag{22}$$

$$\frac{d\tilde{p}}{dt}(t) = \alpha^2 \nabla_q U(\tilde{q}(t)). \tag{23}$$

which defines a flow $\phi(\alpha, t, q, p)$. It should be noted that $\phi(-\alpha, t, q, p) = \phi(\alpha, t, q, p)$ for every α, t, q, p and that ϕ is correctly defined for $\alpha = 0$. In this case, it is easy to see that:

$$\phi(0, t, q, p) = q + pt. \tag{24}$$

As π is the restriction of a positive smooth function, ϕ is defined for every $(\alpha, t, q, p) \in [-1, 1] \times \mathbb{R}^+ \times \mathbb{R}^n \times \mathbb{R}^n$.

Furthermore, $\alpha^2 \nabla_q U$ is smooth. Hence, using the differentiability of the solutions of differential equations on parameters and initial conditions (see [40]), we see that ϕ is C^2 on $]-1, 1[\times Q \times \mathbb{R}^+ \times \mathbb{R}^n$.

Since Q is open, there exists $\rho > 0$ such that $B(0, 2\rho) \subset Q$. Let ν be the measure on Q which is the Lebesgue measure on $B(0, \rho/2)$ and zero outside the ball $B(0, \rho/2)$.

Our aim is to show that there exists $\epsilon > 0$ such that for every $q \in Q$, $P_\pi(q, .) \geq \epsilon \nu(.)$. Note that we would only need that $P_\pi^{n_0}(q, .) \geq \epsilon \nu(.)$ for some n_0, but since Q is convex we will be able to take $n_0 = 1$.

The density of the probability measure associated with momenta is $\exp(-1/2\|p\|^2)$, and taking into account the rescaling of the momenta, we define the probability density

$$\gamma(p) = \alpha \exp(-\frac{1}{2}\|\frac{1}{\alpha}p\|^2). \tag{25}$$

Q is bounded, hence there exists $\epsilon > 0$ such that for every $(q, p) \in \{(q, p) | q \in Q, p \in B(-q, \rho)\}$,

$$\gamma(p) > \epsilon$$

Let

$$A = \{(\alpha, t, q, p) : |\alpha| \leq 1/2, t \in [1/2, 3/2],\ q \in \bar{Q},\ p \in B(-q, \rho)\}.$$

A is compact, hence the first and second order derivatives of ϕ are bounded on A. Thus there exists $Z > 0$ such that for every $(\alpha, t, q, p) \in A$,

$$\|\phi^q(\alpha, t, q, p) - \phi^q(0, 1, q, p)\| \leq (\alpha + |t - 1|)Z \tag{26}$$

and

$$\|d_p \phi^q(\alpha, t, q, p) - d_v \phi^q(0, 1, q, p)\| \leq (\alpha + |t - 1|)Z. \tag{27}$$

Using equation 24, the two above inequalities are equivalent to

$$\|\phi^q(\alpha, t, q, p) - (q + p)\| \leq (\alpha + |t - 1|)Z \tag{28}$$

and

$$\|d_p\phi^q(\alpha, t, q, p) - Id_{\mathbb{R}^n}\| \leq (\alpha + |t - 1|)Z \tag{29}$$

The determinant is continuous, so there exists $0 < \beta < 1$ such that $\|d_p\phi^q(\alpha, t, q, p) - Id_{\mathbb{R}^n}\| < \beta$ implies

$$1/2 < |d_p\phi(\alpha, t, q, p)| < 2. \tag{30}$$

Finally, using Lemma 4 together with the fact that ∇U is bounded, we deduce that there exists $\alpha_0 > 0$ such that for every $0 < \alpha \leq \alpha_0$ and every $q \in Q$ and $p \in B(-q, \rho)$, the trajectory $\phi^q(\alpha, t, q, p)$ for $t \leq 1$ stays in Q. It follows that ϕ and Φ coincide, for there is no reflection.

Let $\alpha = \min(\frac{\rho}{4Z}, \frac{1}{2Z}, \frac{\beta}{2Z}, \alpha_0) > 0$. Let any $t \in [1 - \frac{\beta}{2Z}, 1]$ and q be in Q. By lemma 5 below, the map

$$f : p \to \phi^q(\alpha, t, q, p)$$

is a C^1 diffeomorphism from $B(-q, \rho)$ to $V = \phi^q(\alpha, t, q, B(-q, \rho))$ and by Lemma 6 (applied to f translated by $-q$), $B(0, \rho/2) \subset V$. Furthermore, equation 30 implies that for every $q' \in V$

$$|df^{-1}(q')| > 1/2. \tag{31}$$

Hence, the push forward measure ξ of ν by f is non zero on $B(0, \rho/2)$, and has a density

$$\xi(q') = \nu(f^{-1}(q'))|df^{-1}(q')| \geq \epsilon/2 \tag{32}$$

on $B(0, \rho/2)$. Hence, under the condition that the travel time t is in $[1 - \frac{\beta}{2Z}, 1]$, we get the following transition probability:

$$P(q, .|t \in [1 - \frac{\beta}{2Z}, 1]) \geq \frac{\epsilon}{2}\nu(.)$$

For the random walk, t is sampled uniformly in $[0, 1]$, hence

$$P(q, .) \geq \frac{\epsilon}{2}\frac{\beta}{2Z}\nu(.)$$

which concludes the proof. □

Remark 7. *The previous proof uses $n_0 = 1$. We believe it should be possible to extend the proof to non convex Q by taking $n_0 > 1$, and some extra regularity assumptions on Q.*

6.3 Mixing time for the cube

Lemma 4. We consider the canonical basis of $\mathbb{R}^n$. As shown before in Eq. (10), the trajectory coordinates associated to the Gaussian are all independent from each other. Furthermore, when a reflection with a boundary occurs, it means that one of the coordinates reached 0 or 1. The reflection simply switches the sign of the momentum for this coordinate, leaving other coordinates unchanged. Finally, the initial momentum vector is sampled according to $p(0) \sim \mathcal{N}(0, I_n)$, therefore each coordinate of $p(0)$ is sampled from an independent Gaussian $\mathcal{N}(0,1)$ in $\mathbb{R}$.
Hence we conclude that each coordinate has the behavior of a 1-dimensional HMC random walk sampling a 1-dimensional Gaussian, all independent from each other.

Let us consider the 1-dimensional random walk for a given Gaussian. We write $P_k(x,.)$ the distribution after k steps starting from $x \in [0,1]$, and g the probability density associated with the restriction of the Gaussian to $[0,1]$. Using theorem 3 combined with theorem 1 for the 1-D Gaussian restricted to $[0,1]$, we deduce that there exists $0 < \rho < 1$ such that for all $x \in [0,1]$,

$$\|P_k(x,.) - g\|_{TV} \leq \rho^k.$$

Observe that writing $P_k^{(n)}$ the distribution after k steps in dimension n and g_n the Gaussian restricted to $[0,1]^n$, we have

$$\begin{cases} P_k^{(n)}(x,y) = P_k(x_1,y_1)P_k^{(n-1)}((x_2,...,x_n),(y_2,...,y_n)), \\ g_n(x) = g(x_1)g_{n-1}((x_2,...,x_n)) \end{cases} \tag{33}$$

The total variation distance can be written as

$$\begin{aligned}
\|P_k^{(n)}(x,.) - g_n\|_{TV} &= \frac{1}{2}\int_{[0,1]^n} \left|P^{(n)}(x,y) - g_n(y)\right| dy \\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|P_k(x,y_1)P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|\left(P_k(x,y_1) - g(y_1) + g(y_1)\right)P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&\leq \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|\left(P_k(x,y_1) - g(y_1)\right)P_k^{(n-1)}(x,y_2)\right| dy_1 dy_2 \\
&+ \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|g(y_1)P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&\leq \frac{1}{2}\int_{[0,1]} \left|P_k(x,y_1) - g(y_1)\right| dy_1 \\
&+ \frac{1}{2}\int_{[0,1]^{n-1}} \left|P_k^{(n-1)}(x,y_2) - g_{n-1}(y_2)\right| dy_2
\end{aligned}$$

We deduce

$$\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq n\|P_k(x,.) - g\|_{TV} \leq n\rho^k \tag{34}$$

Hence for a fixed ϵ, if k satisfies $n\rho^k \leq \epsilon$, then for every x, $\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq \epsilon$. Thus we take $k \geq (\log n - \log \epsilon)/(\log 1/\rho)$, and the mixing time is $O(\log n)$.

In addition, as each coordinate is from each other, the total number of reflections is the sum of reflections per coordinate. Hence, the number of reflections is proportional to the dimension. □

7 Supporting information: pseudo-code

In the following, we provide the pseudo-code for the high level description of the HMC algorithm (Algorithm 1).

Algorithm S 1 Hamiltonian Monte Carlo step with real RAM arithmetic model

`HMC_step`(q)
Choose a travel time $L \sim Unif(0, L_{max})$.
choose $p \sim \mathcal{N}(0, I_n)$
Set $dist = L$
while $dist > 0$ **do**
  $(intersection, t_c) \leftarrow$ Intersect_hyper_planes($\mathtt{q}, \mathtt{p}$) // find intersection with hyperplanes
  if intersection = False OR $dist < t_c$ **then**
    (q, p) = Update_positions_momenta$(q, p, dist)$ // update traj. with distance $dist$
    Set $dist = 0$
  else
    (q, p) = Update_positions_momenta(q, p, t_c)
    Reflext_normal$(p(t_c), \mathbf{n}_c)$
    Set $dist = dist - t_c$

Algorithm S 2 Hamiltonian Monte Carlo step with `iRRAM` number type

`HMC_step`(q)
Choose a travel time $L \sim Unif(0, L_{max})$.
choose $p \sim \mathcal{N}(0, I_n)$ `iRRAM` REAL
Set $dist = L$
while $dist > 0$ **do**
  $(intersection, t_c) \leftarrow$ Intersect_hyper_planes(`q`, `p`) with t_c an `iRRAM` REAL.
  $t_c^< =$ Near_inf_double(t_c)
  if intersection = False OR $dist < t_c^<$ **then**
    (q, p) = Update_positions_momenta$(q, p, dist)$ // update trajectory with distance $dist$
  else
    (q, p) = Update_positions_momenta$(q, p, t_c^<)$
    Reflext_normal$(p(t_c^<), \mathbf{n}_c)$
    Set $dist = dist - t_c^<$
  Is_strictly_in_convex[iRRAM :: $Real$](q)
  Return q

Algorithm S 3 Hamiltonian Monte Carlo step with mixed precision. The statistics updated are as follows: $\#R$: multi-precision refinements triggered in `iRRAM`; $\#E$: the number of *exits* of the polytope. See Section 4.3. In this pseudo-code, `CGAL` refers to the numbers types described in section 3.4.4, which are used in a lazy manner using a `try-catch`.

`HMC_step`(q)
Choose a travel time $L \sim Unif(0, L_{max})$.
choose $p \sim \mathcal{N}(0, I_n)$ double precision
$q_{proposed} = HMC\_step < double > (q, p, L)$
if Is_strictly_in_convex[CGAL]($q_{proposed}$) **then**
  Return $q_{proposed}$
else
  increment $\#R$
  $q_{\text{iRRAM}} = HMC\_step <$ iRRAM $> (q, p, L)$
  $q_{proposed} = to\_double(q_{\text{iRRAM}})$
  if Is_strictly_in_convex[CGAL]($q_{proposed}$) **then**
    Return $q_{proposed}$
  else
    increment $\#E$
    Return q

Algorithm S 4 Find trajectory parameters for a given direction.

Compute_traj_params(q_{dir}, p_{dir})
Set $\omega = \sqrt{2a}$
Compute $C = \sqrt{q_{dir}^2 + p_{dir}^2/w^2}$
Compute $\phi = \arctan(-\frac{p_{dir}}{q_{dir}\omega})$
if $p_{dir} < 0$ and $\phi < 0$ **then**
    $\phi = \phi + \pi$
if $p_{dir} > 0$ and $\phi > 0$ **then**
    $\phi = \phi - \pi$
Return $[\omega, C, \phi]$

Algorithm S 5 Intersecting the trajectory with hyperplanes bounding the polytope

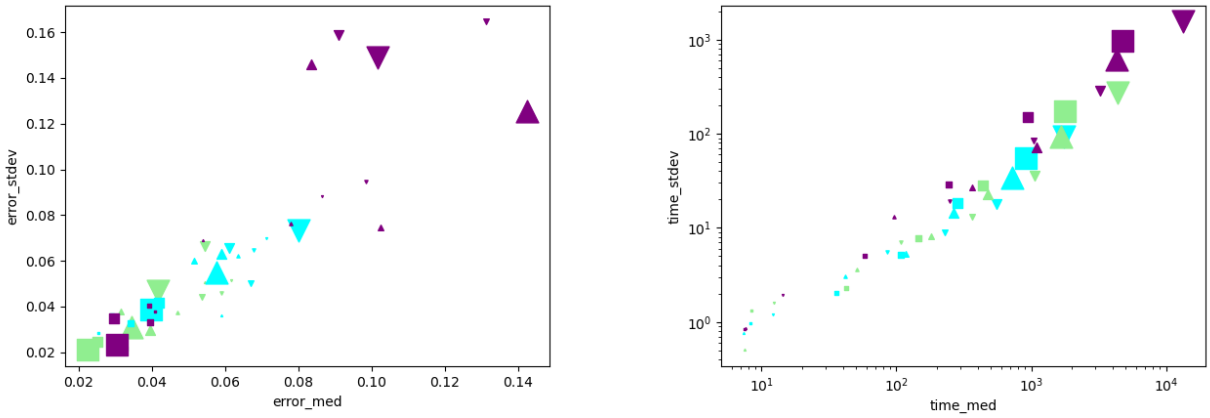
```
Intersect_hyper_planes(q, p)
Set intersection = False
for each hyperplane H of equation (Ax)_i = b_i do
  Compute the outward pointing normal n_H to the hyper-plane
  Compute the dot products q_{n_H} =< q, n_H > and p_{n_H} =< p, n_H >
  [ω, C, Φ] = Compute_traj_params(q_{n_H}, p_{n_H})
  if C > b_i then
    t1 = (arccos(b_i/C) − φ) /ω
    if t1 < 0 then
      t1 = t1 + 2π/ω
    t2 = (− arccos(b/C) − φ) /ω
    if t2 < 0 then
      t2 = t2 + 2π/ω
    t = min(t1, t2)
    if intersection = False then
      Set t_c = t
      Set t_c = n_H
      Set intersection = True
    else
      if t < t_c then
        Set t_c = t
        Set n_c = n_H
Return (intersection, t_c)
```

Algorithm S 6 Reflecting the normal

Reflext_normal(p, n)
$n' = n/\|n\|$ // unit normal
Return $p - 2 < p, n' > n'$

Algorithm S 7 Update trajectory with distance t

1: `Update_positions_momenta`(q, p, t)
2: **for** i from 1 to n **do**
3: $[\omega, C, \Phi] =$ `Compute_traj_params` (q_i, p_i)
4: Set $q_i = C\cos(\omega t + \phi)$
5: Set $p_i = -\omega C \sin(\omega t + \phi)$
6: Return (q, p)



$median(\text{err}_r) \times stdev(\text{err}_r)$ $median(time) \times stdev(time)$

Figure S 1: **Estimating the volume of three polytopes with known volumes (cube, isotropic simplex, standard simplex), in dimension $n = 10, 30, 50, 70, 100$, with three algorithms.** Statistics reported over 50 runs. The error of an estimate $\tilde{V}$ is defined as $\text{err}_r = |\tilde{V} - V|/V$. **Symbol shape vs polytope:** square: cube; triangle up: isotropic simplex; triangle down: standard simplex **Symbol size vs dimension :** smallest for $n = 10$, largest for $n = 100$ **Color vs algorithm:** purple/light blue/light green HAR/HMC0/HMC1.5

8 Supporting information: results

8.1 Up to dimension $n = 100$

N	Algo.	model	n	ε	Vol	$\min \tilde{V}$	$\max \tilde{V}$	$med(\tilde{V})$	$stdev(\tilde{Vol})$	$med(\mathrm{err}_r)$	$stdev(\mathrm{err}_r)$
50	HAR-Wa500-Wb4.0-Wc2.0	cube	10	0.01	1.024E+03	9.240E+02	1.181E+03	1.017E+03	6.396E+01	4.127E-02	3.804E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	10	0.01	1.024E+03	9.084E+02	1.135E+03	1.041E+03	4.365E+01	2.545E-02	2.833E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	10	0.01	1.024E+03	9.296E+02	1.112E+03	1.007E+03	4.137E+01	2.934E-02	2.475E-02
50	HAR-Wa500-Wb4.0-Wc2.0	cube	30	0.01	1.074E+09	9.438E+08	1.260E+09	1.085E+09	6.812E+07	3.977E-02	4.033E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	30	0.01	1.074E+09	9.222E+08	1.194E+09	1.056E+09	5.889E+07	2.913E-02	3.598E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	30	0.01	1.074E+09	9.903E+08	1.166E+09	1.075E+09	4.381E+07	3.406E-02	2.160E-02
50	HAR-Wa500-Wb4.0-Wc2.0	cube	50	0.01	1.126E+15	9.828E+14	1.237E+15	1.112E+15	6.447E+13	4.050E-02	3.307E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	50	0.01	1.126E+15	9.973E+14	1.315E+15	1.119E+15	5.945E+13	3.563E-02	3.272E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	50	0.01	1.126E+15	1.035E+15	1.242E+15	1.124E+15	4.417E+13	2.363E-02	2.409E-02
50	HAR-Wa500-Wb4.0-Wc2.0	cube	70	0.01	1.181E+21	1.080E+21	1.408E+21	1.175E+21	6.071E+19	2.993E-02	3.489E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	70	0.01	1.181E+21	1.013E+21	1.407E+21	1.169E+21	7.877E+19	4.311E-02	4.187E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	70	0.01	1.181E+21	1.080E+21	1.296E+21	1.196E+21	4.854E+19	2.595E-02	2.471E-02
50	HAR-Wa500-Wb4.0-Wc2.0	cube	100	0.01	1.268E+30	1.147E+30	1.385E+30	1.268E+30	5.240E+28	3.081E-02	2.335E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	100	0.01	1.268E+30	1.103E+30	1.436E+30	1.271E+30	7.623E+28	4.221E-02	3.852E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	100	0.01	1.268E+30	1.163E+30	1.361E+30	1.256E+30	4.071E+28	2.274E-02	2.102E-02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	10	0.01	1.472E+04	1.208E+04	1.984E+04	1.468E+04	1.456E+03	5.587E-02	6.912E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	10	0.01	1.472E+04	1.285E+04	1.686E+04	1.447E+04	1.076E+03	6.128E-02	3.624E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	10	0.01	1.472E+04	1.155E+04	1.810E+04	1.417E+04	1.304E+03	5.609E-02	5.054E-02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	30	0.01	7.067E+12	5.307E+12	9.046E+12	6.867E+12	8.776E+11	8.019E-02	7.639E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	30	0.01	7.067E+12	5.218E+12	8.329E+12	6.914E+12	6.941E+11	6.996E-02	6.234E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	30	0.01	7.067E+12	6.090E+12	8.154E+12	6.948E+12	4.534E+11	4.734E-02	3.738E-02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	50	0.01	3.421E+21	2.369E+21	4.213E+21	3.225E+21	4.277E+20	1.025E-01	7.448E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	50	0.01	3.421E+21	2.574E+21	4.220E+21	3.400E+21	3.089E+20	5.191E-02	5.987E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	50	0.01	3.421E+21	3.045E+21	3.960E+21	3.408E+21	2.056E+20	3.181E-02	3.765E-02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	70	0.01	1.658E+30	1.074E+30	2.910E+30	1.581E+30	3.286E+29	8.505E-02	1.461E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	70	0.01	1.658E+30	1.311E+30	2.091E+30	1.645E+30	1.667E+29	5.953E-02	6.289E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	70	0.01	1.658E+30	1.489E+30	1.924E+30	1.677E+30	9.053E+28	4.052E-02	2.953E-02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	100	0.01	1.771E+43	9.292E+42	2.731E+43	1.691E+43	3.321E+42	1.473E-01	1.257E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	100	0.01	1.771E+43	1.416E+43	2.106E+43	1.732E+43	1.556E+42	5.782E-02	5.497E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	100	0.01	1.771E+43	1.528E+43	2.004E+43	1.773E+43	8.912E+41	3.469E-02	3.094E-02
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	10	0.01	2.756E-07	2.101E-07	4.015E-07	2.639E-07	3.616E-08	8.853E-02	8.825E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	10	0.01	2.756E-07	2.079E-07	3.443E-07	2.703E-07	3.035E-08	7.123E-02	6.970E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	10	0.01	2.756E-07	2.332E-07	3.314E-07	2.703E-07	2.382E-08	6.167E-02	5.142E-02
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	30	0.01	3.770E-33	2.530E-33	5.840E-33	3.578E-33	5.488E-34	1.011E-01	9.483E-02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	30	0.01	3.770E-33	2.804E-33	4.818E-33	3.648E-33	3.909E-34	6.972E-02	6.489E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	30	0.01	3.770E-33	3.097E-33	4.326E-33	3.760E-33	2.961E-34	6.082E-02	4.581E-02
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	50	0.01	3.288E-65	1.388E-65	6.294E-65	3.182E-65	7.545E-66	1.354E-01	1.646E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	50	0.01	3.288E-65	2.554E-65	3.813E-65	3.136E-65	2.739E-66	6.691E-02	5.023E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	50	0.01	3.288E-65	2.687E-65	3.816E-65	3.215E-65	2.544E-66	5.715E-02	4.430E-02
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	70	0.01	8.348E-101	4.621E-101	1.434E-100	7.952E-101	1.796E-101	9.139E-02	1.589E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	70	0.01	8.348E-101	6.633E-101	1.147E-100	8.207E-101	8.571E-102	6.389E-02	6.583E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	70	0.01	8.348E-101	7.052E-101	1.195E-100	8.397E-101	7.726E-102	5.556E-02	6.656E-02
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	100	0.01	1.072E-158	6.380E-159	2.022E-158	1.039E-158	2.317E-159	1.020E-01	1.491E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	100	0.01	1.072E-158	7.746E-159	1.446E-158	1.052E-158	1.264E-159	8.130E-02	7.348E-02
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	100	0.01	1.072E-158	9.210E-159	1.382E-158	1.056E-158	7.451E-160	4.200E-02	4.704E-02

Table S 1: **Statistics on volumes and their estimates.** See text for details.

N	Algo.	model	n	ε	med($\#S$)	stdev($\#S$)	med($\#O$)	stdev($\#O$)	med($\#R$)	stdev($\#R$)	med($\#E$)	stdev($\#E$)	med(time)	stdev(time)
50	HAR-Wa500-Wb4.0-Wc2.0	cube	10	0.01	2.238E+05	1.740E+04	NA	NA	NA	NA	NA	NA	7.450E+00	8.461E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	10	0.01	6.142E+04	4.080E+03	7.214E+04	5.253E+03	0	1.979E-01	0	0.000E+00	8.323E+00	9.835E-01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	10	0.01	6.492E+04	4.849E+03	7.680E+04	6.022E+03	0	1.414E-01	0	0.000E+00	8.426E+00	1.341E+00
50	HAR-Wa500-Wb4.0-Wc2.0	cube	30	0.01	2.109E+06	1.059E+05	NA	NA	NA	NA	NA	NA	5.831E+01	5.098E+00
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	30	0.01	1.388E+05	6.275E+03	2.226E+05	1.153E+04	0	5.067E-01	0	0.000E+00	3.590E+01	2.021E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	30	0.01	1.970E+05	9.190E+03	3.133E+05	1.785E+04	0	4.870E-01	0	0.000E+00	4.226E+01	2.272E+00
50	HAR-Wa500-Wb4.0-Wc2.0	cube	50	0.01	7.113E+06	2.791E+05	NA	NA	NA	NA	NA	NA	2.461E+02	2.862E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	50	0.01	2.025E+05	9.241E+03	4.129E+05	2.266E+04	1	1.020E+00	0	0.000E+00	1.076E+02	5.157E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	50	0.01	3.692E+05	1.748E+04	7.481E+05	4.242E+04	1	8.631E-01	0	0.000E+00	1.452E+02	7.777E+00
50	HAR-Wa500-Wb4.0-Wc2.0	cube	70	0.01	1.607E+07	7.101E+05	NA	NA	NA	NA	NA	NA	9.533E+02	1.489E+02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	70	0.01	2.686E+05	1.119E+04	6.596E+05	3.936E+04	2	1.256E+00	0	0.000E+00	2.856E+02	1.850E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	70	0.01	6.053E+05	2.341E+04	1.497E+06	7.134E+04	3	1.588E+00	0	0.000E+00	4.412E+02	2.796E+01
50	HAR-Wa500-Wb4.0-Wc2.0	cube	100	0.01	3.905E+07	1.251E+06	NA	NA	NA	NA	NA	NA	4.858E+03	9.663E+02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	100	0.01	3.513E+05	1.469E+04	1.090E+06	5.353E+04	3	1.858E+00	0	0.000E+00	9.122E+02	5.562E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	cube	100	0.01	1.030E+06	4.072E+04	3.204E+06	1.303E+05	7	2.830E+00	0	0.000E+00	1.765E+03	1.745E+02
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	10	0.01	3.700E+05	2.860E+04	NA	NA	NA	NA	NA	NA	7.644E+00	8.724E-01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	10	0.01	1.053E+05	7.105E+03	1.331E+05	9.633E+03	0	3.405E-01	0	0.000E+00	7.302E+00	7.681E-01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	10	0.01	1.129E+05	7.718E+03	1.429E+05	1.045E+04	0	1.414E-01	0	0.000E+00	7.469E+00	5.051E-01
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	30	0.01	3.512E+06	2.124E+05	NA	NA	NA	NA	NA	NA	9.596E+01	1.304E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	30	0.01	2.472E+05	1.233E+04	4.679E+05	3.005E+04	0	7.624E-01	0	0.000E+00	4.203E+01	3.073E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	30	0.01	3.632E+05	1.694E+04	6.773E+05	3.821E+04	1	8.533E-01	0	0.000E+00	5.079E+01	3.588E+00
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	50	0.01	1.209E+07	6.025E+05	NA	NA	NA	NA	NA	NA	3.622E+02	2.705E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	50	0.01	3.672E+05	1.499E+04	9.210E+05	4.995E+04	1	1.265E+00	0	0.000E+00	1.164E+02	5.273E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	50	0.01	7.124E+05	2.902E+04	1.867E+06	9.848E+04	2	1.301E+00	0	0.000E+00	1.798E+02	8.136E+00
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	70	0.01	2.824E+07	1.365E+06	NA	NA	NA	NA	NA	NA	1.089E+03	7.212E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	70	0.01	4.842E+05	1.794E+04	1.517E+06	1.025E+05	4	1.876E+00	0	0.000E+00	2.641E+02	1.444E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	70	0.01	1.175E+06	4.792E+04	3.861E+06	1.839E+05	7	2.740E+00	0	0.000E+00	4.801E+02	2.295E+01
50	HAR-Wa500-Wb4.0-Wc2.0	isotropic-Δ	100	0.01	6.836E+07	3.077E+06	NA	NA	NA	NA	NA	NA	4.351E+03	6.187E+02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	100	0.01	6.316E+05	2.290E+04	2.562E+06	1.322E+05	9	3.144E+00	0	0.000E+00	7.285E+02	3.410E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	isotropic-Δ	100	0.01	2.017E+06	8.785E+04	8.885E+06	4.817E+05	24	3.750E+00	0	0.000E+00	1.671E+03	9.442E+01
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	10	0.01	7.797E+05	4.155E+04	NA	NA	NA	NA	NA	NA	1.434E+01	1.961E+00
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	10	0.01	2.167E+05	1.287E+04	2.593E+05	1.668E+04	0	9.750E-01	0	0.000E+00	1.208E+01	1.200E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	10	0.01	2.350E+05	1.445E+04	2.837E+05	1.937E+04	1	1.035E+00	0	0.000E+00	1.241E+01	1.618E+00
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	30	0.01	9.288E+06	3.183E+05	NA	NA	NA	NA	NA	NA	2.489E+02	1.948E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	30	0.01	6.575E+05	2.216E+04	1.013E+06	4.502E+04	21	6.122E+00	0	0.000E+00	8.526E+01	5.515E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	30	0.01	9.321E+05	3.788E+04	1.474E+06	7.733E+04	30	6.531E+00	0	0.000E+00	1.075E+02	7.122E+00
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	50	0.01	3.585E+07	1.194E+06	NA	NA	NA	NA	NA	NA	1.042E+03	8.565E+01
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	50	0.01	1.014E+06	2.499E+04	1.881E+06	7.828E+04	119	1.556E+01	0	0.000E+00	2.302E+02	9.022E+00
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	50	0.01	1.905E+06	4.713E+04	3.730E+06	1.535E+05	205	2.136E+01	0	0.000E+00	3.655E+02	1.324E+01
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	70	0.01	8.629E+07	2.611E+06	NA	NA	NA	NA	NA	NA	3.233E+03	2.901E+02
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	70	0.01	1.457E+06	2.554E+04	3.083E+06	1.055E+05	379	3.030E+01	0	0.000E+00	5.492E+02	1.795E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	70	0.01	3.402E+06	6.912E+04	7.760E+06	2.721E+05	813	6.718E+01	0	0.000E+00	1.063E+03	3.572E+01
50	HAR-Wa500-Wb4.0-Wc2.0	standard-Δ	100	0.01	2.174E+08	5.125E+06	NA	NA	NA	NA	NA	NA	1.348E+04	1.551E+03
50	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	100	0.01	2.044E+06	4.459E+04	5.157E+06	2.014E+05	1319	9.533E+01	0	0.000E+00	1.755E+03	9.324E+01
50	HMC-Wa250-Wb1.0-Wc1.5-r0.1	standard-Δ	100	0.01	6.203E+06	1.274E+05	1.746E+07	6.092E+05	3914	2.482E+02	0	0.000E+00	4.424E+03	2.704E+02

Table S 2: **Misc statistics.** The columns read as follows: $\#S$: num sampled points. $\#O$: num calls to the oracle. $\#R$: multi-precision refinements triggered in `iRRAM`. $\#E$: the number of *exits* of the polytope.

8.2 Dimensions $n = 250, 500$

N	Algo.	model	n	ε	Vol	$\min \check{V}$	$\max \check{V}$	$med(\check{V})$	$stdev(\check{V}ol)$	$med(\mathrm{err}_r)$	$stdev(\mathrm{err}_r)$
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	250	0.05	1.809E+75	1.463E+75	2.285E+75	1.817E+75	3.463E+74	1.886E-01	1.037E-01
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	250	0.05	NAN	1.306E+108	2.849E+108	2.445E+108	6.030E+107	NAN	NAN
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	250	0.05	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	-1.000E+00	0.000E+00

Table S 3: **Statistics on volumes and their estimates.** See text for details.

N	Algo.	model	n	ε	med($\#S$)	stdev($\#S$)	med($\#O$)	stdev($\#O$)	med($\#R$)	stdev($\#R$)	med($\#E$)	stdev($\#E$)	med(time)	stdev(time)
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	250	0.05	1.596E+05	2.891E+03	1.011E+06	5.436E+04	11	4.604E+00	0	0.000E+00	4.279E+03	4.374E+01
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	isotropic-Δ	250	0.05	3.024E+05	5.553E+03	2.659E+06	9.444E+04	25	5.586E+00	0	0.000E+00	4.699E+03	6.557E+01
5	HMC-Wa250-Wb0.0-Wc0.0-r0.1	standard-Δ	250	0.05	1.030E+06	1.637E+04	4.709E+06	8.287E+04	8083	1.856E+02	0	0.000E+00	2.353E+04	5.417E+02

Table S 4: **Misc statistics.** The columns read as follows: $\#S$: num sampled points. $\#O$: num calls to the oracle. $\#R$: multi-precision refinements triggered in `iRRAM`. $\#E$: the number of *exits* of the polytope.

N	Algo.	model	n	ε	Vol	$\min \check{V}$	$\max \check{V}$	$med(\check{V})$	$stdev(\check{V}ol)$	$med(\mathrm{err}_r)$	$stdev(\mathrm{err}_r)$
3	HMC-Wa250-Wb0.0-Wc0.0-r0.01	cube	500	0.05	3.273E+150	1.067E+150	2.463E+150	1.463E+150	7.193E+149	5.529E-01	2.197E-01
3	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	500	0.05	3.273E+150	2.835E+150	3.655E+150	3.226E+150	4.100E+149	1.165E-01	6.451E-02

Table S 5: **Statistics on volumes and their estimates.** See text for details.

N	Algo.	model	n	ε	med($\#S$)	stdev($\#S$)	med($\#O$)	stdev($\#O$)	med($\#R$)	stdev($\#R$)	med($\#E$)	stdev($\#E$)	med(time)	stdev(time)
3	HMC-Wa250-Wb0.0-Wc0.0-r0.01	cube	500	0.05	3.635E+05	3.570E+03	8.563E+05	5.624E+04	8	3.786E+00	0	0.000E+00	9.262E+03	3.183E+02
3	HMC-Wa250-Wb0.0-Wc0.0-r0.1	cube	500	0.05	2.619E+05	8.789E+03	3.350E+06	1.325E+05	59	4.000E+00	0	0.000E+00	3.366E+04	1.027E+03

Table S 6: **Misc statistics.** The columns read as follows: $\#S$: num sampled points. $\#O$: num calls to the oracle. $\#R$: multi-precision refinements triggered in `iRRAM`. $\#E$: the number of *exits* of the polytope.

Contents